Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



**Звіт**

З лабораторної роботи № 7

З дисципліни «Об’єктно-орієнтоване програмування»

**На тему**: «Множинне спадкування. Поліморфізм»

**Виконав:** ст. гр. КІ-15

Колосенко Д.О.

**Прийняв:**

Козак Н.Б.

Львів 2020

**Мета:** познайомитися із множинним спадкуванням класів та поліморфізмом.

**Теоретичні відомості**

Якщо спадкування здійснюється від декількох батьківських класів одночасно, тоді воно називається *множинним спадкуванням*. Визначальним для похідного класу породженого множинним спадкуванням є те, що він явно чи неявно повинен успадковувати характеристики декількох базових класів.

Основні принципи одинарного спадкування, зокрема спадкування членів, модифікаторів доступу до членів базових класів, розширення та обмеження характеристик, без жодних доповнень можуть бути перенесені на множинне спадкування.

*Неявним множинним спадкуванням* можуть бути випадки змішаного спадкування. Результатом цих спадкуваннь є ієрархія, в якій похідний клас неявно (через один проміжний) успадкував характеристики двох базових класів. Якщо похідний клас породжується від декількох базових, то в декларації класу треба вказати усі базові класи, розділяючи їх комою, разом зі специфікаторами спадкування.

*Порядок передавання аргументів конструкторам* в оголошенні конструктора похідного класу може бути довільним, оскільки виклики і виконання конструкторів визначаються порядком спадкування в оголошенні класу. *Порядок виклику деструкторів є таким як у конструкторів, а виконання - зворотнім*.

Поліморфізм – це здатність коду при постійному інтерфейсі змінювати свою поведінку в залежності від ситуації, яка виникає на момент виконання. Іншими словами один і той же метод може бути визначений для об‘єктів різних класів, що є між собою в ієрархії спадкування, при цьому метод якого класу викликати вирішується під час виконання програми.

*Статичний поліморфізм* реалізовується за допомогою так званого *раннього зв’язування* через механізм перевантаження функцій, методів та операторів і віртуальні класи.

*Динамічний поліморфізм* реалізовується за допомогою так званого *пізнього зв’язування* через механізм віртуальних функцій.

**Завдання:**

Спроектувати і реалізувати ієрархію класів, що описують предметну область згідно варіанту, яка реалізується класом 1. Клас 1 в свою чергу утворюється шляхом множинного спадкування класів 2 і 3 кожен з яких в свою чергу успадковує клас 4. Додаткові вимоги:

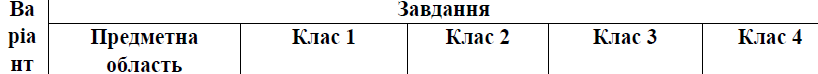
1. Базовий клас містить мінімум один віртуальний метод, один невіртуальний метод і одну динамічно створювану властивість.

2. Забезпечити механізми коректної роботи конструкторів і деструкторів.

3. Перевантажити оператор присвоєня з метою його коректної роботи.

4. Кожен з класів має містити мінімум одину властивість і 4 методи.

5. Написати main() функцію де створити об‘єкт класу 1 і продемонструвати різницю між статичним і динамічним поліморфізмом.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **14** | Розкладний стіл | CFoldingTable | CCommode | CTable | CFurniture |

**Код програми:**

**Main**

#include <iostream>

#include "Furniture.h"

using namespace std;

int main()

{

СFurniture a1("Luks", 250, 750);

СFurniture a11("Luks2", 200, 150);

CTable a2("MEH245", 25, 500, true);

CCommode a3("MEH245", 350, 400, true);

CFoldingTable a4("SV", "KM45", 545, 65, true, false);

a1.ShowInfo();

a11.ShowInfo();

a11 = a1;

a1.ShowInfo();

a2.ShowInfo();

a2.OpenClose();

a3.ShowInfo();

a3.OpenClose();

a4.ShowInfo();

СFurniture\* a20 = new CTable("LL", 145, 220, true);

СFurniture\* a30 = new CCommode("ZZ", 35, 40, false);

CFoldingTable a40("Sl", "Kl45", 855, 605, true, false);;

СFurniture\* a40 = new CFoldingTable("Sl", "Kl45", 855, 605, true, false);

a20->ShowInfo();

a20->ShowInfo();

a30->ShowInfo();

a40->ShowInfo();

system("pause");

return 0;

}

**Furniture.h**

#pragma once

#include <iostream>

#include <time.h>

using namespace std;

class СFurniture

{

protected:

string Furniture\_Name;

double width, height;

const int size=10;

int\* dynMasuv;

public:

СFurniture();

СFurniture(int w, int h);

СFurniture(string name, int w, int h);

virtual ~СFurniture();

double GetWidth();

double GetHeight();

virtual void ShowInfo();

СFurniture& operator=(const СFurniture&);

};

**Furniture.cpp**

#include "Furniture.h"

СFurniture::СFurniture()

{

Furniture\_Name = "No name";

width = 0;

height = 0;

dynMasuv = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuv[i] = rand() % 10 + 1;

}

}

СFurniture::СFurniture(string name, int w, int h)

{

Furniture\_Name = name;

width = w;

height = h;

dynMasuv = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuv[i] = rand() % 10 + 1;

}

}

СFurniture::СFurniture(int w, int h)

{

Furniture\_Name = "No name";

width = w;

height = h;

dynMasuv = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuv[i] = rand() % 10 + 1;

}

}

СFurniture::~СFurniture()

{

delete[] dynMasuv;

}

double СFurniture::GetHeight()

{

return height;

}

double СFurniture::GetWidth()

{

return width;

}

void СFurniture::ShowInfo()

{

cout << "\n---Info---\n" << "Name: " << Furniture\_Name << "\nWidth: " << width << "\nHeight: " << height << "\nRandom: ";

for (int i = 0; i < size; i++)

{

cout << dynMasuv[i] << " ";

}

cout << endl;

}

СFurniture& СFurniture::operator=(const СFurniture& obj)

{

Furniture\_Name = obj.Furniture\_Name;

width = obj.width;

height = obj.height;

for (int i = 0; i < size; i++)

{

dynMasuv[i] = obj.dynMasuv[i];

}

return \*this;

}

**Table.h**

#pragma once

#include <iostream>

#include "Furniture.h"

using namespace std;

class CTable : public СFurniture

{

protected:

string Table\_Model;

bool folding;

int\* dynMasuvTable;

public:

CTable();

CTable(string Table\_Model, int h, int w, bool ch);

void OpenClose();

virtual ~CTable();

virtual void ShowInfo();

};

**Table.ccp**

#include "Table.h"

CTable::CTable():СFurniture()

{

Table\_Model = "No model";

folding = false;

dynMasuvTable = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuvTable[i] = rand() % 10 + 1;

}

}

CTable::CTable(string model, int h, int w, bool ch): СFurniture(h,w)

{

Table\_Model = model;

folding = ch;

dynMasuvTable = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuvTable[i] = rand() % 10 + 1;

}

}

void CTable::OpenClose()

{

if (folding == true)

{

cout << "\nTable is folding";

}

else

{

cout << "\nTable is`nt folding";

}

}

CTable::~CTable()

{

delete[] dynMasuvTable;

}

void CTable::ShowInfo()

{

cout << "\n---Info---\n" << "Name: " << Furniture\_Name<<"\nModel: "<< Table\_Model << "\nWidth: " << width << "\nHeight: " << height << "\nRandom: ";

for (int i = 0; i < size; i++)

{

cout << dynMasuvTable[i] << " ";

}

cout << endl;

}

**Commode.h**

#pragma once

#include <iostream>

#include "Furniture.h"

using namespace std;

class CCommode : public СFurniture

{

protected:

string Comomd\_Model;

bool OC;

int\* dynMasuvCommod;

public:

CCommode();

CCommode(string Commod\_Model, int h, int w, bool ch);

void OpenClose();

virtual ~CCommode();

virtual void ShowInfo();

};

**Commode.ccp**

#include "Commode.h"

CCommode::CCommode() :СFurniture()

{

Comomd\_Model = "No model";

OC = false;

dynMasuvCommod = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuvCommod[i] = rand() % 10 + 1;

}

}

CCommode::CCommode(string model, int h, int w, bool ch) : СFurniture(h, w)

{

Comomd\_Model = model;

OC = ch;

dynMasuvCommod = new int[size];

srand(time(0));

for (int i = 0; i < size; i++)

{

dynMasuvCommod[i] = rand() % 10 + 1;

}

}

void CCommode::OpenClose()

{

if (OC == true)

{

cout << "\nCommode is open";

}

else

{

cout << "\nComode is close";

}

}

CCommode::~CCommode()

{

delete[] dynMasuvCommod;

}

void CCommode::ShowInfo()

{

cout << "\n---Info---\n" << "Name: " << Furniture\_Name << "\nModel: " << Comomd\_Model << "\nWidth: " << width << "\nHeight: " << height << "\nRandom: ";

for (int i = 0; i < size; i++)

{

cout << dynMasuvCommod[i] << " ";

}

cout << endl;

}

**FoldingTable.h**

#pragma once

#include <iostream>

#include "Commode.h"

#include "Table.h"

using namespace std;

class CFoldingTable : public CCommode, public CTable

{

public:

CFoldingTable();

CFoldingTable(string model, string model2, int h, int w, bool ch, bool ch2);

virtual ~CFoldingTable();

virtual void ShowInfo();

};

**FoldingTable.cpp**

#include "FoldingTable.h"

CFoldingTable::CFoldingTable() : CCommode(), CTable()

{

Comomd\_Model = "No model";

Table\_Model = "No model";

OC = false;

folding = false;

}

CFoldingTable::CFoldingTable(string model, string model2, int h, int w, bool ch, bool ch2) : CTable(model,h,w,ch), CCommode(model2, h, w, ch2){ }

CFoldingTable::~CFoldingTable(){}

void CFoldingTable::ShowInfo()

{

cout << "\n---Info---\n" << "Name: " << Furniture\_Name << "\nTable Model: " << Table\_Model << "\nCommode Model: " << Comomd\_Model << "\nWidth: " << width << "\nHeight: " << height << "\nRandom: ";

for (int i = 0; i < size; i++)

{

cout << dynMasuvCommod[i] << " ";

}

cout << endl << "Random2: ";

for (int i = 0; i < size; i++)

{

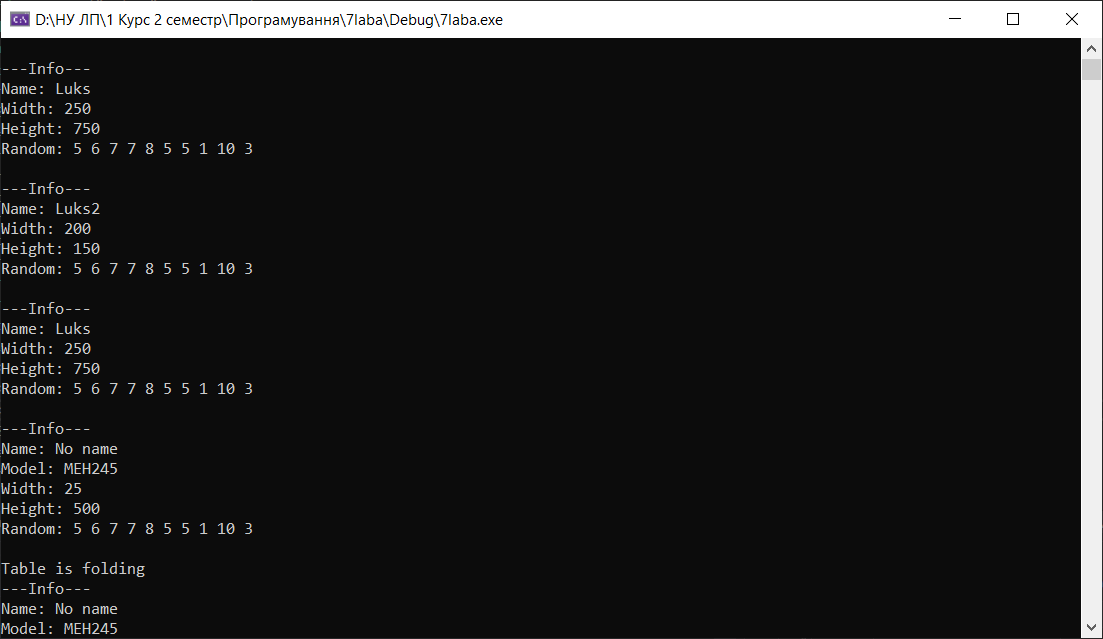
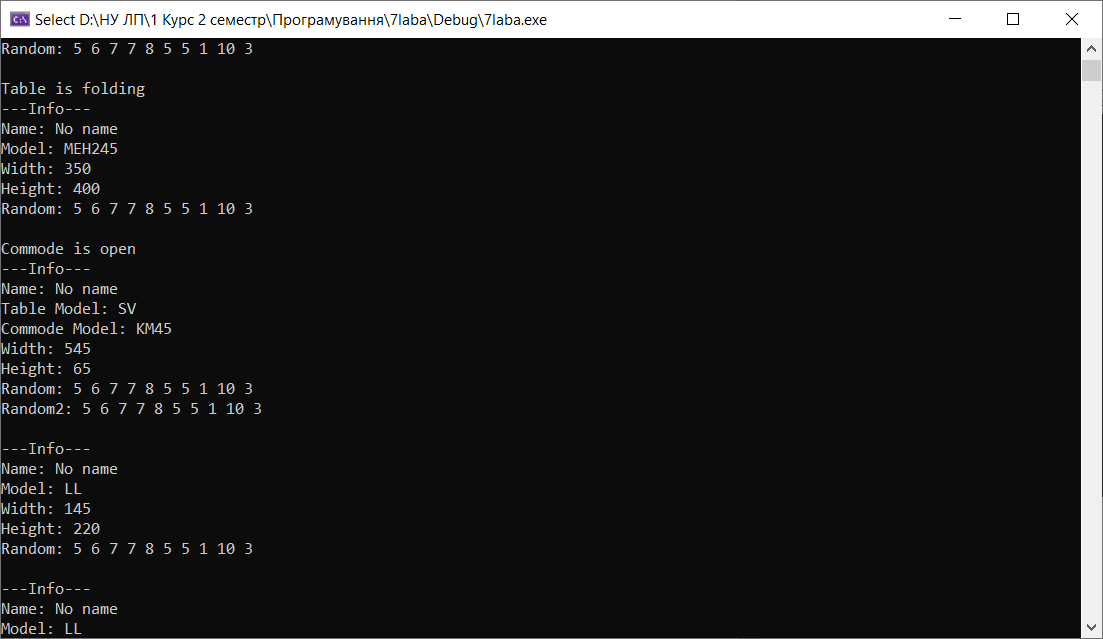
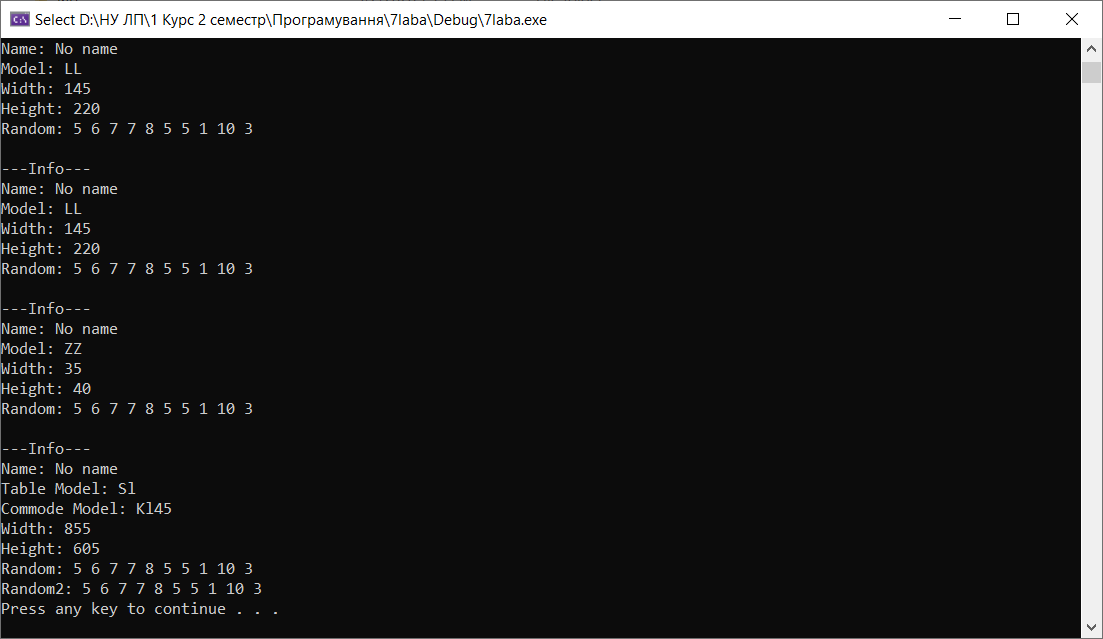
cout << dynMasuvTable[i] << " ";

}

cout << endl;

}

**Результат програми:**

**Висновок:** Під час виконання цієї лабораторної роботи я познайомився із множинним спадкуванням класів та поліморфізмом.